CAD-PROGRAMM

CARLOS

HANDBUCH FÜR BENUTZER

Hans-Jörg Mathony
Institut für Technik der Informationsverarbeitung
Universität Karlsruhe

CARLOS - BENUTZERHANDBUCH

1. Einführung 1

1.1 CARLOS als Bestandteil des Programmsystems FIGARO 1
1.2 Schaltnetz- und Steuerwerksentwurf mit CARLOS 4
1.3 Gliederung des Handbuchs 4

2. Installation und Benutzung von CARLOS 6

2.1 CARLOS-Dateien 6
2.2 Eingabe-Syntax 6
2.2.1 Sequentielle Logik 8
2.2.2 Kombinatorische Logik 15

3. CAR_TM: Synthese von zweistufigen Schaltnetzen 16

3.1 Entwurfsablauf 16
3.2 Synthesemodi und Optimierungskriterien 18
3.3 Ausgabe von CAR_TM 19
3.4 Fehlermeldungen 20

4. CAR_MS: Synthese von mehrstufigen CMOS-Schaltnetzen 22

4.1 Entwurfsablauf 22
4.2 Gatterbibliothek und Technologieparameter 25
4.3 Steuerung der Synthese 30
4.4 Ausgabe von CAR_MS 34
4.4.1 Netzliste 34
4.4.2 Statistische Daten 37
4.5 Fehlermeldungen 38

5. Literatur zu CARLOS 39

1. Einführung

Das CAD-Programm **CARLOS** entstand in den Jahren 1984 bis 1987 im Rahmen des Forschungs- und Entwicklungsprojekts **MEGA** (**M**odulares **E**ntwurfssystem für **G**ate-**A**rrays) am Institut für Technik der Informationsverarbeitung der Universität Karlsruhe.

CARLOS ermöglicht den automatischen logischen Entwurf von Schaltnetzen und Schaltwerken in CMOS-Technologie. Es können sowohl zweistufige UND/ODER-Schaltnetze (PLA's und PAL's) als auch mehrstufige CMOS-Schaltnetze synthetisiert werden, wobei der Benutzer durch Vorgabe von Stufenzahl, Optimierungsmodi und -kriterien die Synthese und deren Ergebnis stark beeinflussen kann.
Als Bestandteil des größeren CAD-Systems **FIGARO** (**F**ully **I**ntegrated **G**ate-**A**rray **O**ptimizer) synthetisiert CARLOS nach erfolgter, automatischer Zustandskodierung Steuerwerke, die auf realiserungsunabhängiger Ebene durch Ablaufdiagramme beschrieben werden.

An der Entwicklung von CARLOS haben Diplomanden und wissenschaftliche Hilfskräfte tatkräftig mitgeholfen, namentlich Frau H. Dücker und die Herren T. Müller, O. Rettig und V. D. Sanchez.

Das vorliegende Handbuch vermittelt alle Kenntnisse, die für die Bedienung des Programms von Wichtigkeit sind.
Leser, die darüberhinaus Informationen oder Details über die implementierten Algorithmen haben möchten, finden in Kapitel 5 weiterführende Literatur.

1.1 CARLOS als Bestandteil des Programmsystems FIGARO

FIGARO wurde im Rahmen des Gemeinschaftsprojekts MEGA von zwei Instituten der Universität Karlsruhe, dem Institut für Technik der Informationsverarbeitung (Logikentwurf) und dem Institut für Theoretische Elektrotechnik und Messtechnik (Physikalischer Entwurf) entwickelt.
Als CAD-System der 3. Generation unterstützt FIGARO sowohl den physikalischen Entwurf auf der Layoutebene (automatische Plazierung und Verdrahtung) als auch den Logikentwurf auf höherer,

systemnaher Ebene durch entsprechende CAD-Programme.
Während die Layout-Algorithmen speziell für Gate-Arrays ausgelegt sind, sind die Ergebnisse der Logiksynthese-Algorithmen auch für andere Realisierungsformen, wie PLA, Standardzellen und vollkundenspezifische Schaltungen verwertbar.
Die Entwurfsmethodik, die FIGARO zugrundeliegt, geht aus Bild 1.1 hervor.

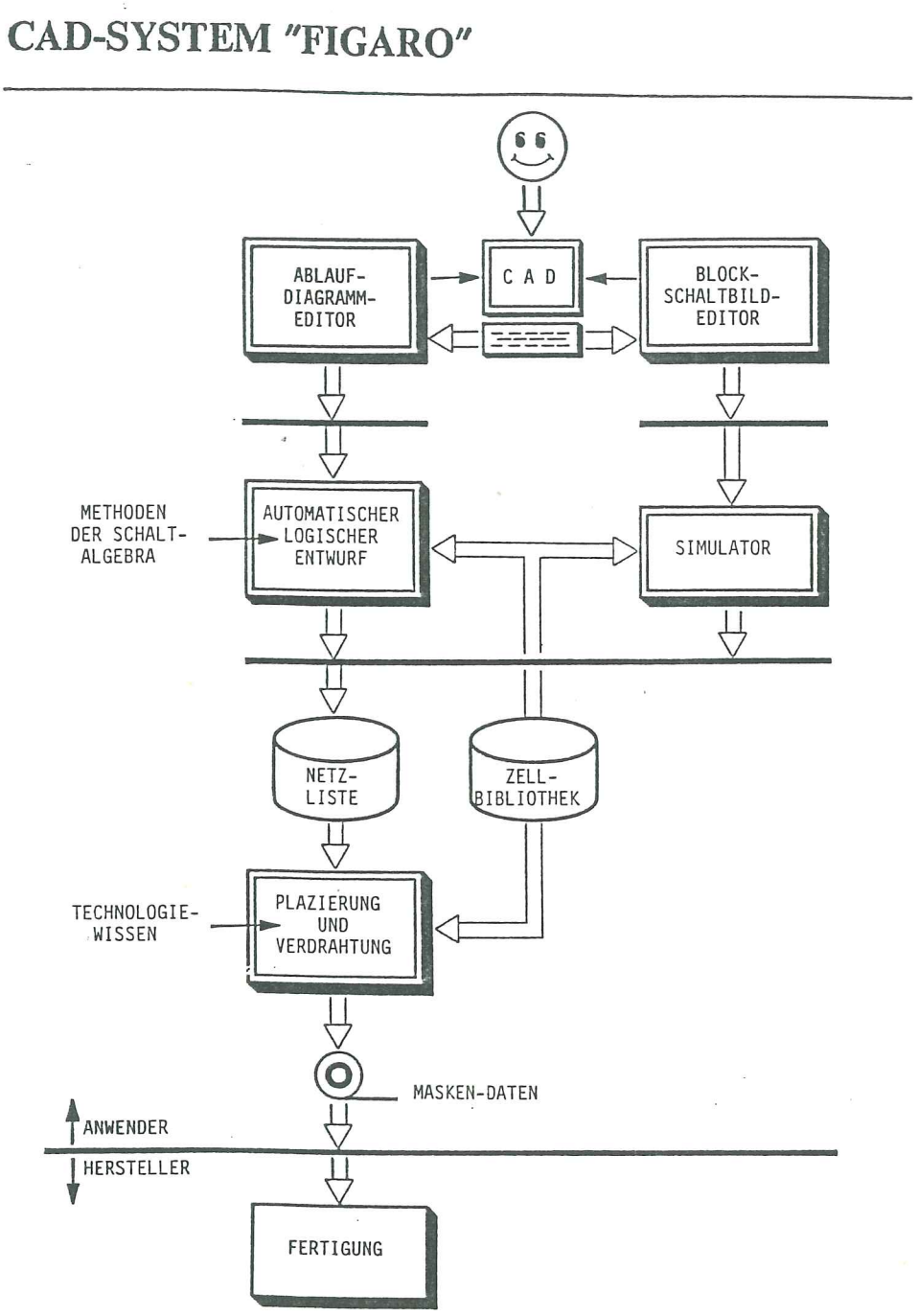


Bild 1.1: Entwurfsablauf des CAD-Systems FIGARO

Dem Systementwickler stehen, entsprechend der klassischen Aufteilung einer Digitalschaltung in **Steuerwerk** und **Operationswerk**, zwei Graphik-Editoren zur Spezifikation seiner Schaltung zur Verfügung: Durch anschauliche Eingabe eines **Ablaufdiagramms** wird das Verhalten des Steuerwerks und durch hierachisch gliederte **Blockschaltbilder** die Struktur des Operationswerks beschrieben.

Für das intuitiv entworfene Operationswerk (OPW) wird nach erfolgter Syntaxprüfung gleich die Netzliste generiert, die Ausgangsbasis für die logische Simulation ist und als Schnittstelle für den physikalischen Entwurf dient.

Die Weiterverabeitung der Ablaufdiagramm-Daten für die Synthese der Logikstruktur des Steuerwerks (STW) erfolgt in mehreren Schritten:
Nach einer Syntaxprüfung wird das Ablaufdiagramm in eine **Ablauftabelle** übersetzt. Die Ablauftabelle bildet die Schnittstelle für die eigentliche Logiksynthese. Zunächst werden den symbolischen Zuständen in der Ablauftabelle binäre Codewörter automatisch zugewiesen, und zwar derart, daß der Realisierungsaufwand der Schaltung möglichst klein wird. Nach der **Zustandskodierung** liegt die Beschreibung der Steuerwerksfunktionen (Überführungs- und Ausgabefunktionen) in Form einer **Bündelfunktionstabelle** vor. Ausgehend von dieser Funktionstabelle und weiteren Informationen, wie z. B. Flipflop-Typ und Grundzustand, wird unter Berücksichtigung der für die CMOS-Technologie typischen Einschränkungen (begrenztes Fan-In und Fan-Out, Gattertyp), die Struktur des Steuerwerks automatisch durch das CAD-Programm CARLOS synthetisiert.
Das Gesamtergebnis des logischen Entwurfs (OPW und STW) wird in einer gemeinsamen Netzliste abgespeichert und an den automatischen physikalischen Entwurf weitergereicht.

1.2 Schaltnetz- und Steuerwerksentwurf mit CARLOS

Das CAD-Programm CARLOS besteht aus den zwei Modulen **CAR_TM** und **CAR_MS.**

Bild 1.2 zeigt den Syntheseablauf im Überblick. Ausgehend von der Funktionstabelle der zu realisierenden Schaltfunktion erfolgt der Entwurf in drei Schritten: Zunächst wird durch logische Minimierung ein (näherungsweise) kostenminimales zweistufiges Bündelschaltnetz synthetisiert, das dann im zweiten Schritt unter Einhaltung der technologischen und anwenderspezifischen Nebenbedingungen in ein optimales mehrstufiges Schaltnetz, bestehend aus UND-Gattern, ODER-Gattern und Invertern, umgesetzt wird. Im dritten Schritt wird das so gewonnenne Schaltnetz transformiert und damit in eine technologisch günstige Form gebracht.

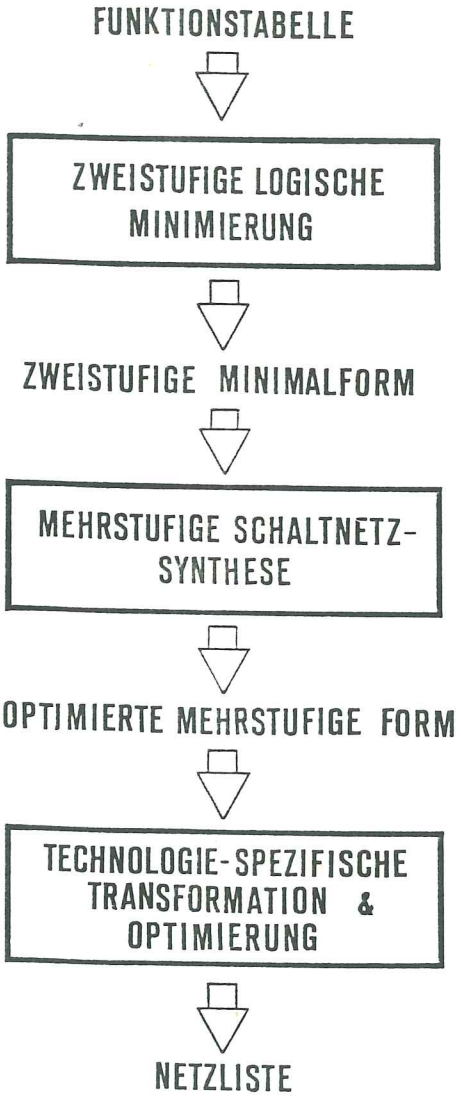


Bild 1.2: Entwurfsablauf für Schaltnetze

Bei der Synthese von Steuerwerken wird von der klassischen Schaltwerksstruktur (kombinatorisches Schaltnetz + Zustandsregister, s. Bild 2.2) ausgegangen, wobei für die Speicherung des Zustandes wahlweise D- oder JK-Flipflops zur Verfügung stehen.
Die Aufteilung der Synthese auf die Module CAR_TM und CAR_MS ist wie folgt:

CAR_TM: **zweistufige logische Minimierung**
Eingabe: Funktionstabelle + weitere Daten (ASCII-Format)
Ausgabe: Strukturtabelle + weitere Daten (ASCII-Format)

CAR_MS: **mehrstufige Synthese** und **Transformation**
Eingabe: Ausgabe von CAR_TM, Gatterbibliothek
Ausgabe: Netzliste, statistische Daten

1.3 Gliederung des Handbuchs

Das vorliegende Benutzer-Handbuch ist wie folgt gegliedert. Auf die Einführung, die Sie gerade lesen, folgt in **Kapitel 2** die Anleitung zur Installation von CARLOS auf ihrem Rechner. Neben den CARLOS-Dateien beschreibt dieses Kapitel auch die Eingabesyntax für die Spezikation von Schaltfunktionen. **Kapitel 3** behandelt das Modul **CAR_TM** für die Synthese von zweistufigen Schaltnetzen, gibt Auskunft über die verschiedenen Optimierungsmodi und schließt mit einem Beispiel.
Das **Kapitel 4** beschreibt die mehrstufige Schaltnetzsynthese, die das Modul **CAR_MS** leistet. Neben einem Überblick über den Entwurfsablauf und der Darstellung der technologiespezifischen Eingabeparameter erfährt der Benutzer, wie er den Syntheseablauf beeinflussen kann, um das für ihn optimale Schaltnetz zu gewinnen.
Der Leser, der über den reinen Gebrauch des Programms mehr wissen möchte über CARLOS, findet in **Kapitel 5** ein dokumentierte Literaturliste.

2. Installation und Benutzung von CARLOS

2.1 CARLOS-Dateien

Auf dem CARLOS-Magnetband befinden sich die folgenden Dateien:

CAR_TM.EXE : Synthese von zweistufigen Schaltnetzen
CAR_MS.EXE : Synthese von mehrstufigen CMOS-Schaltnetzen

CARLOS.COM : CARLOS-Kommandoprozedur

GATLIB.DAT : Gatterbibliothek, Technologiespezif. Daten

<NAME>.FUN : Beispiel-Datensätze für CAR_TM
<NAME>.STR : Beispiel-Datensätze für CAR_MS

Erzeugt werden folgende Dateien:

<NAME>.STA : CAR_MS-Ausgabedatei, statistische Daten
<NAME>.NET : CAR_MS-Ausgabedatei, Netzliste

Der Zusammenhang zwischen ausführbaren Programmen und Datensätzen geht aus dem Programm-Ablauf in Bild 2.1 hervor. Die programminternen logischen Namen für Ein- und Ausgabedateien sind darin *kursiv* eingetragen.

2.2 Eingabe-Syntax

Die Eingabesyntax von CARLOS entspricht weitgehend der Eingabesprache des CAD-Programms LOGE (ISDATA GmbH, Karlsruhe).

Die Eingabedaten sind in Bereiche eingeteilt, die durch bestimmte Schlüsselwörter gekennzeichnet sind. Schlüsselwörter werden durch das Zeichen '*' in der ersten Spalte markiert.

Die Datensätze für CAR_TM und CAR_MS unterscheiden sich nur
- in der Art der Funktionsbeschreibung und
- in den Optionen, die man zur Steuerung des jeweiligen Programms angeben kann.

Für die Synthese von sequentieller Logik können die Eingabedaten nach graphischer Eingabe des Ablaufdiagramms mit Hilfe des FIGA-RO-Steuerwerkseditors automatisch generiert werden.
Im folgenden wird die Dateneingabe für sequentielle Logik beschrieben. Die Behandlung von kombinatorischer Logik kann als Sonderfall betrachtet werden; die Unterschiede in der Eingabe werden in Abschnitt 2.2.2 beschrieben.

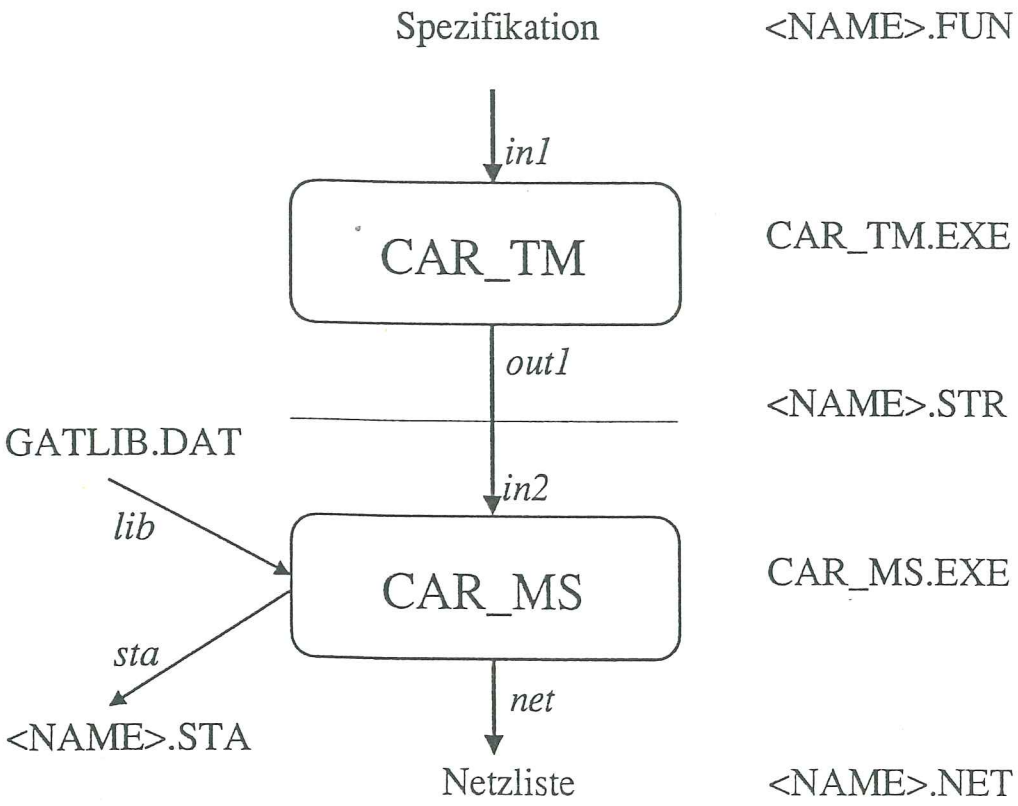


Bild 2.1: CARLOS - Programmablauf

2.2.1 Sequentielle Logik

CARLOS unterstützt den Entwurf von synchronen Steuerwerken mit klassischer Grundstruktur (Bild 2.2). Der Zustand des Steuerwerks wird in einem Register aus synchron getakteten Flipflops abgespeichert. Folgezustands- und Steuervariablen werden in Abhängigkeit des jeweiligen Zustandes und vom Wert der Entscheidungsvariablen durch den kombinatorischen Teil der Schaltung (Überführungs- und Ausgabeschaltnetz) generiert.
Zur Beschreibung eines Steuerablaufs kann die **Ablauftabelle** verwendet werden. Sie geht durch eine 1:1-Transformation aus dem Ablaufdiagramm hervor.
Die Ablauftabelle enthält eingangsseitig den

- **Zustand** und die
- **Werte der Entscheidungsvariablen**

und ausgangsseitig den

- **Folgezustand** und die
- **Werte der Steuervariablen.**

Jede Zeile der Ablauftabelle beschreibt somit einen Zustandsübergang des Steuerwerks.

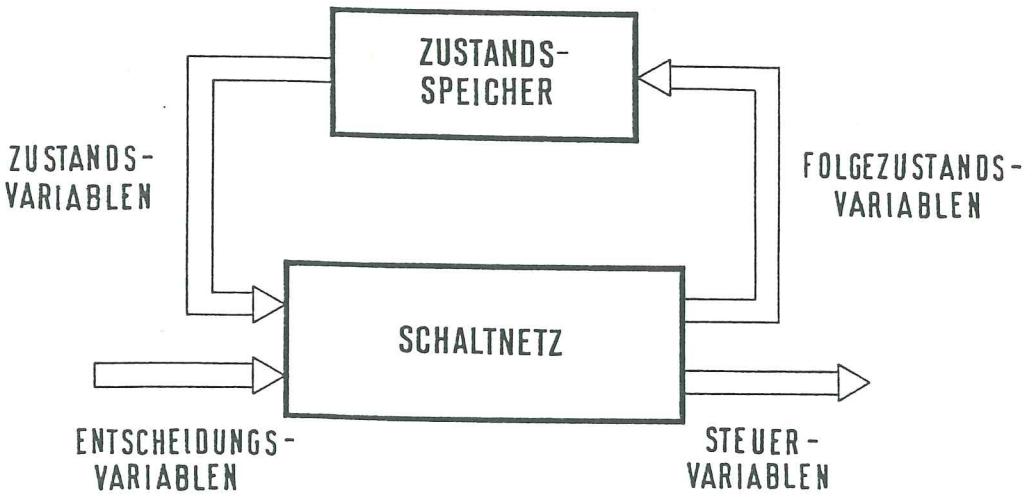


Bild 2.2: Grundstruktur des klassischen Steuerwerks

Für (Folge-)Zustände trägt man deren symbolischen Namen in die Ablauftabelle ein. Die Entscheidungsvariablen erhalten binäre Werte ('0'oder '1') bzw. den Eintrag '-', falls die Variable für den entsprechenden Zustandsübergang nicht relevant ist.
Den Steuervariablen werden die Werte '0' oder '1' bzw. das Zeichen '-' zugeordnet, wobei '-' bedeutet, daß die entsprechende Funktion für die Eingangskombination nicht spezifiziert ist.

Durch die Codierung der Zustände des STW's entsteht die **codierte Ablauftabelle.** Die codierte Ablauftabelle wird in Abhängigkeit des jeweiligen Flipfloptyps in eine **Bündelfunktionstabelle** transformiert, aus der die Spezifikation der Überführungs- und Ausgangsfunktionen eindeutig hervorgeht.

Beispiel: symbolische und codierte Ablauftabelle
4 Zustände, 3 Entscheidungs- und 3 Steuervariablen

S1, 0-- ; S1, 000	00 0-- ; 00 000
S1, 10- ; S2, 0-1	00 10- ; 01 0-1
S1, 11- ; S3, 100	00 11- ; 10 0-1
S2, 00- ; S2, 0-1	01 00- ; 01 0-1
S2, 1-- ; S3, 100	01 1-- ; 10 100
S3, -0- ; S1, 000	10 -0- ; 00 000
S3, -1- : S4, 111	10 -1- ; 11 111
S4, --0 ; S1, 000	11 --0 ; 00 000
S4, -01 ; S2, 0-1	11 -01 ; 01 0-1
S4, -11 ; S4, 111	11 -11 ; 11 111
symbolisch	**codiert**

CARLOS akzeptiert als Eingabe nur die Funktionstabelle der Überführungs- und Ausgabefunktionen eines Steuerwerks. Symbolische Ablauftabellen können mit dem FIGARO-Modul **CASTRO** automatisch in Funktionstabellen umgesetzt werden. Die Zustandszuweisung erfolgt dabei derart, daß der Realisierungsaufwand für die Schaltnetze möglichst klein wird.

Für die Synthese von Steuerwerken (Schaltnetz + getaktete Flip-flops) müssen die folgenden Schlüsselwörter in einem CARLOS-Eingabedatensatz enthalten sein.

***IDENTIFICATION** dient zur Identifikation des Entwurfs. Es werden max. 40 Zeichen gelesen.
Bis zum nächsten Schlüsselwort werden alle Daten als Kommentar interpretiert.

Beispiel:

```
*IDENTIFICATION
Bus-Arbiter Steuerwerk - 13.03.88
```

***DECLARATIONS** dient zur Angabe der Anzahl von Ein- und Ausgangsvariablen

X-VAR = Anzahl der Entscheidungsvariablen
Y-VAR = Anzahl der Ausgangsvariablen
QX-VAR = Anzahl der Zustandsvariablen
QY-VAR = Anzahl der Flipflop-Ansteuervariablen

Beispiel: (STW mit JK-Flipflops)

```
*DECLARATIONS
 X-VAR = 7
 Y-VAR = 3
 QX-VAR = 3
 QY-VAR = 6
```

Die korrespondierende Schaltfunktion hat 10 (X-VAR + QX-VAR) Ein-Eingangs- und 9 (Y-VAR + QY-VAR) Ausgangsvariablen.

***X-NAMES** Datenbereich für die Namen der Eingangsvariablen
***Y-NAMES** Datenbereich für die Namen der Ausgangsvariablen

Die Namen dürfen aus maximal 7 Buchstaben bestehen. Die Nummer, die nach dem Gleichheitszeichen folgt, gibt die Position der zugehörigen Variablen in der Funktionstabelle an, wobei von links nach rechts gezählt wird.
Für Zustands- und FF-Ansteuervariablen müssen keine Namen angegeben werden.

Beispiel:

```
*X-NAMES
 SIN = 5, TO = 6, STO = 7, RGB = 8, ...
*Y-NAMES
 TOUT = 5, Z = 6, SET = 7, ...
```

Man beachte, daß in der Funktionstabelle eingangs- und ausgangsseitig die ersten Spalten jeweils den Zustands- bzw. den FF-Ansteuervariablen zugeordnet sind.

***OPTIONS** ermöglicht die Eingabe von Parametern zur Beeinflussung der Synthese. Genaue Beschreibung in den Abschnitten 3.2 und 4.3 (Eingabe nicht notwendig).

***FUNCTION-TABLE** definiert die zu realisierende Schaltfunktion in tabellarischer Form.

Diese Eingabe ist für CAR_TM erforderlich. CAR_TM erzeugt daraus die für CAR_MS notwendige Eingabe in Form einer **Strukturtabelle.**

Die Funktionstabelle beschreibt in kompakter Form die Werte der Ausgabefunktionen Y_k in Abhängigkeit der Wertekombinationen für den Eingangsvektor X (X umfaßt dabei Zustands- und Entscheidungsvariablen; Y umfaßt FF-Ansteuer- und Ausgabevariablen).
Eingangskombinationen können blockweise aufgeführt werden. Nicht aufgeführte Eingangsbelegungen werden automatisch der 'don't-care'-Menge zugeordnet.

Eine Zeile der Funktionstabelle hat die folgende Form

Eingangsseite ; **Ausgangsseite**

$\mathring{q}_1\ \mathring{q}_2 \ldots \mathring{q}_p\ \mathring{x}_1\ \mathring{x}_2 \ldots \mathring{x}_n$; $\mathring{Q}_1\ \mathring{Q}_2 \ldots \mathring{Q}_r\ \mathring{Y}_1\ \mathring{Y}_2 \ldots \mathring{Y}_m$

wobei p = QX-VAR
n = X-VAR
r = QY-VAR
m = Y-VAR

Im Eingangsteil der Funktionstabelle treten folgende Komponenten auf:

$\mathring{q}_i$, $\mathring{x}_i$ =
0: Variable negiert
1: Variable bejaht
-: Variable sowohl bejaht als auch negiert

Entsprechend gilt im Ausgabeteil:

$\mathring{Q}_k$, $\mathring{Y}_k$ =
0: Funktion Q_k (Y_k) nimmt den Wert 0 an
1: Funktion Q_k (Y_k) nimmt den Wert 1 an
-: Funktion Q_k (Y_k) ist für die entsprechende Belegung nicht spezifiziert

Die Komponenten der Funktionstabelle dürfen in Gruppen duch 'Blanks' getrennt angegeben werden.

Beispiel:

```
*FUNCTION-TABLE
 000 01- --- , 101 0-10
 000 000 --- , 100 0101
 000 001 01- , 000 0-0-
 000 001 1-- , 111 110-
 ...
```

Die Beschreibung der Funktionstabelle endet mit dem nächsten Schlüsselwort.

***FLIPFLOPS** legt den Flipfloptyp fest; unterstützt werden D- und JK-Flipflops (Eingabe nicht notwendig; Voreinstellung: D-Flipflops).

***PRESET** definiert den Grundzustand (binäres Codewort), in den das Steuerwerk geht, wenn das Signal PRESET den logischen Wert '1' annimmt.

Beispiel:

```
*FLIPFLOPS
 D-FF
*PRESET
 1010
```

***END** bildet den Abschluß des Eingabedatensatzes.

Die Reihenfolge der Datenbereiche ist beliebig, mit einer Ausnahme: Der Deklarationsteil (*DECLARATIONS) muß vor der Definition der Namen und der Funktionstabelle auftreten.
Abschließend ein vollständiger Datensatz.

Beispiel:

```
*IDENTIFICATION
 Steuerwerk eines Pufferspeichers (Klausuraufgabe WS 1984)
*DECLARATIONS
 X-VAR = 3
 Y-VAR = 6
 QX-VAR = 3
 QY-VAR = 3
*X-NAMES
 S = 4, L = 5, G = 6
*Y-NAMES
 FS = 4, FL = 5, A = 6, FR = 7, LR = 8, VL = 9
*FLIPFLOP
 D-FF
*PRESET
 000
*FUNCTION-TABLE
 000 101 , 001 00-010
 000 0-1 , 000 00-010
 001 00- , 010 000100
 010 000 , 100 10-000
 010 001 , 011 10-000
 011 -01 , 011 00-001
 011 011 , 101 00-001
 100 000 , 100 00-000
 100 100 , 001 00-000
 100 010 , 101 00-000
 101 00- , 110 001000
 110 000 , 100 01-000
 110 001 , 000 01-000
*END
```

2.2.2 Kombinatorische Logik

Die Dateneingabe für kombinatorische Logik unterscheidet sich nur geringfügig von der Eingabe für sequentielle Logik.
Die nur für Steuerwerke relevanten Schlüsselwörter *FLIPFLOPS und *PRESET werden weggelassen. Die Parameter QX-VAR und QY-VAR zur Festlegung der Anzahl der Zustands- und Flipflopansteuervariablen werden gleich 0 gesetzt oder ebenfalls einfach weggelassen.

Beispiel: Eingabedatensatz für kombinatorische Logik

```
*IDENTIFICATION
 Full Adder
*DECLARATIONS
 X-VAR = 4
 Y-VAR = 3
*X-NAMES
 Ai = 1, Bi = 2, Ci-1 = 3
*Y-NAMES
 Si = 1, Ci = 2
*FUNCTION-TABLE
 000 , 00
 001 , 10
 010 , 10
 011 , 01
 100 , 10
 101 , 01
 110 , 01
 111 , 11
*END
```

QX-VAR = ∅
QY-VAR = ∅

4

Sonst Stack-Dump

3. CAR_TM: Synthese von zweistufigen Schaltnetzen

3.1 Entwurfsablauf

Der Minimierer CAR_TM synthetisert ausgehend von der Funktionstabelle einer Schaltfunktion ein optimales zweistufiges Schaltnetz in disjunktiver UND/ODER- (NAND/NAND-) Struktur.
Die Schaltnetzkosten werden im wesentlichen durch drei Parameter bestimmt:

1) Anzahl der UND-Gatter in der Eingangsstufe
2) Anzahl der Gattereingänge in der Eingangsstufe
3) Anzahl der Gattereingänge in der Ausgangsstufe.

Die Anzahl der ODER-Gatter (in der Ausgangsstufe) ist identisch mit der Anzahl der Funktionen eines Bündels und damit bei gegebener Problemstellung fest.

Die schaltalgebraische Darstellung der vollständigen Schaltfunktion, die ein zweistufiges UND/ODER-Schaltnetz beschreibt, ist durch die folgenden Parameter gekennzeichnet:

1) Anzahl der Produktterme
2) Summe der Produktterm-Literale
3) Summe der Produktterm-Ausnutzungen

CAR_TM berücksichtigt bei der Minimierung alle drei Kostenparameter. Mit Hilfe von Optionen, die weiter unten beschrieben werden, kann der Optimierungsschwerpunkt (Anzahl der Gatter oder Anzahl der Gattereingänge) festgelegt werden. Unabhängig vom Optimierungskriterium ist das resultierende Schaltnetz irredundant, d. h. es enthält keine überflüssigen Gatter oder Verbindungen).
Ausgabe von CAR_TM ist die **Strukturtabelle** des synthetisierten Schaltnetzes (Beschreibung in Abschnitt 3.3).
Bei Eingabe einer entsprechenden Option berechnet CAR_TM zusätzlich redundante Verbindungen von Gattern der Eingansgsstufe zu Gattern der Ausgangsstufe und kennzeichnet diese in der Strukturtabelle. Diese redundanten Verbindungen stellen Frei-

heitsgrade dar, die bei der Umsetzung des zweistufigen Schaltnetzes in ein mehrstufiges Schaltnetz ausgenutzt werden können.

Die Vorgehensweise des Minimierers CAR_TM ist an die Strategie der bekannten Minimierer **MINI** und **ESPRESSO-II** angelehnt.
Die Minimierung basiert auf der iterativen Wiederholung der folgenden Prozeduren:

0) **Cons_Merge** - Implikanten werden zu kürzeren Implikanten zusammengefaßt.

1) **Prim_Expand** - Implikanten werden zu Primimplikanten expandiert.

2) **Ess_Primes** - Bestimmung der Kernimplikanten.

3) **Irr_Cover** - Berechnung einer irredundanten Überdeckung.

4) **Reduce** - Primimplikanten werden maximal 'verlängert'.

5) **Low_Outs** - Die Anzahl der Ausnutzungen von Primimplikanten wird minimiert.

6) **Exp_Inp** - Implikanten, die nicht prim sind, werden bei fest vorgegebener Ausnutzung nochmals expandiert.

7) **Exp_Out** - Die maximale Ausnutzung eines Primimplikanten wird berechnet.

Welche Prozeduren wie oft und in welcher Reihenfolge durchlaufen werden hängt von dem jeweiligen Synthesemodus ab.

CAR_TM überprüft die Eingabe auf Konsistenz, d. h. auf Widersprüchlichkeiten in der Spezifikation der Schaltfunktion, und verifiziert das Syntheseergebnis nach abgeschlossener Minimierung durch Vergleich mit den Eingabedaten. Dabei auftretende Fehler werden am Bildschirm ausgegeben (s. Abschnitt 3.4).

3.2 Synthesemodi und Optimierungskriterium

Durch Eingabe von Optionen kann der Minimierungsprozeß und dessen Ergebnis beeinflußt werden.

***OPTIONS**

SINGLE - Einzelminimierungsmodus: kein Produktterm wird mehrfach ausgenutzt.

TURBO - nicht-iterative, sehr schnelle Minimierung.

MIN_CON - iterative Minimierung der Anzahl von Produktterm-ausnutzungen (empfiehlt sich, wenn man die Anzahl der Gattereingänge minimieren will; wirkt sich günstig auf die anschließende mehrstufige Synthese aus).

RED_CON - in diesem Modus werden zusätzlich redundante Verbindungen im Schaltnetz berechnet und mit der Strukturtabelle ausgegeben (s. Abschnitt 3.3).

TRACE - der Syntheseablauf wird am Bildschirm dokumentiert.

Optionen können in Klein- oder Großbuchstaben durch Komma getrennt angegeben werden.

Wenn der Benutzer keine Optionen angibt, sind die obigen Parameter alle auf "FALSE" gesetzt.

Tips für die Eingabe von Optionen:

Schaltnetz	Optionen
PAL	SINGLE und TURBO
PLA	TURBO und MIN_CON
mehrstufig	TURBO und MIN_CON, SINGLE und TURBO, SINGLE und TURBO und MIN_CON

Wenn die Option TURBO nicht gesetzt ist, läuft der Minimierer im iterativen "Normal-Modus". Das Ergebnis ist dann meistens absolut minimal (auch hinsichtlich der Anzahl der Gattereingänge, wenn die Option MIN_CON gesetzt ist), oder sehr nahe am exakten Minimum. Die Laufzeit im Normal-Modus ist im Durchschnitt 3 bis 5 mal so lang wie im TURBO-Mode.

3.3 Ausgabe von CAR_TM

Abgesehen von dem Datenbereich, der die zu synthetisierende Schaltfunktion definiert, ist die Ausgabe von CAR_TM identisch mit der Eingabe.
Anstelle der Schaltfunktion gibt das Programm unter dem Schlüsselwort ***STRUCTURE-TABLE** die Strukturtabelle des ermittelten Schaltnetzes aus.
Die Zeilen der Strukturtabelle sind den UND- (NAND-) Gattern der Eingangstufe im korrespondierenden Schaltnetz zugeordnet. Die eingangsseitigen Komponenten einer Zeile beschreiben die Beschaltung der Gattereingänge, die Komponenten im Ausgabeteil der Strukturtabelle beschreiben die Ausnutzung der Gatter in der Ausgangsstufe des Schaltnetzes.

Die Komponenten im Eingangsteil der Zeile j können die folgenden Werte annehmen:

	0:	Eingangssignal i wird **bejaht** ...
Komponente i =	**1:**	Eingansgsignal i wird **negiert** ...
	-:	Eingangssignal i wird **nicht** ...

... an das UND-Gatter j angeschlossen.

Im Ausgangsteil haben die Werte der Komponenten der Zeile j folgende Bedeutung:

	0:	Ausgangssignal Gatter j **darf nicht** ...
Komponente k =	**1:**	Ausgangssignal Gatter j **muß** ...
	d:	Ausgangssignal Gatter j **darf** ...

... an das ODER-Gatter k angeschlossen werden.

Die Strukturtabelle wird durch das nächste Schlüsselwort abgeschlossen.

Beispiel:

```
*DECLARATIONS
 X-VAR = 3
 Y-VAR = 3
*STRUCTURE-TABLE
 1-0 , 110
 11- , 1d1
 -10 , 011
 0-0 , 001
*END
```

Strukturtabelle

x_1	x_2	x_3	Y_1	Y_2	Y_3
1	-	0	1	1	0
1	1	-	1	d	1
-	1	0	0	1	1
0	-	0	0	0	1

Eingangsteil | Ausgangsteil

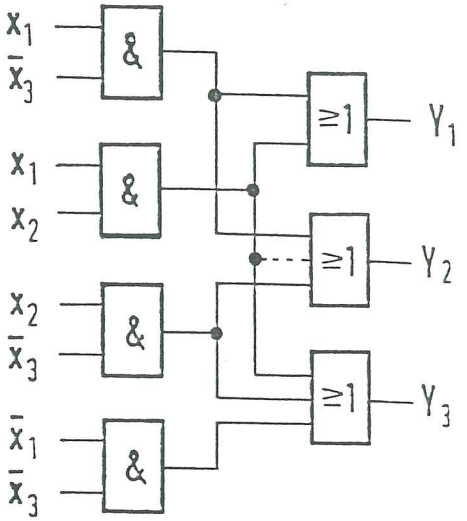


Bild 3.1: Strukturtabelle und Schaltnetz

3.4 Fehlermeldungen

CAR_TM überprüft die Spezifikation einer Schaltfunktion auf Widerspruchsfreiheit. Wenn für eine Wertekombination der Eingangsvariablen zwei unterschiedliche Funktionswerte angegeben sind, so reagiert CAR_TM mit der Fehlermeldung

"ERROR inconsistent lines - check your function table"

und bricht den Syntheseablauf ab.

Nach der Minimierung der Schaltfunktion wird durch Vergleich des Ergebnisses mit der ursprünglichen Spezifikation eine **Verifikation** durchgeführt. Dabei wird überprüft, ob

1) jede Einsstelle und
2) **keine** Nullstelle der zu ursprünglichen Schaltfunktion

von der minimierten Schaltfunktion überdeckt wird. Entsprechend dieser Überprüfung können zwei Fehlermeldungen auftreten:

1) **"ERROR: cube ... is NOT totally covered"**
2) **"ERROR: cube ... intersects OFF-set".**

Diese Fehlermeldungen deuten auf einen Programmfehler hin. Bitte informieren Sie umgehend den Wartungsdienst!

4. CAR_MS: Synthese von mehrstufigen CMOS-Schaltnetzen

4.1 Entwurfsablauf - Überblick

Das Programm CAR_MS synthetisiert ausgehend von der Strukturtabelle eines zweistufigen Schaltnetzes (Ausgabe von CAR_TM) ein global optimiertes mehrstufiges CMOS-Schaltnetz.
Ausgabe von CAR_MS ist eine

- **Netzliste**, aus der die Struktur des mehrstufigen Schaltnetzes eindeutig hervorgeht (bei sequentiellen Schaltungen enthält die Netzliste die vollständige Beschreibung der Schaltwerksstruktur) und eine

- Datei mit **statistischen Daten**, wie Angaben über Stufenzahl, Laufzeiten, Fan-Out-Belastungen der Schnittstellen- und internen Signale, Gatter-Statistik usw.

CAR_MS erzeugt grundsätzlich ein gemeinsames Schaltnetz (Bündelschaltnetz) für alle m Funktionen, unabhängig davon, ob die Strukturtabelle der Eingabedatei einzeln minimierte Funktionen oder eine im Bündel minimerte Funktion darstellt.

Die Synthese, deren Ablauf Bild 4.1 zeigt, ist in zwei Phasen geliedert:

Phase 1 leistet die Umsetzung des zweistufigen Schaltnetzes in ein optimales mehrstufiges Schaltnetz, bestehend aus UND-Gattern, ODER-Gattern und Invertern.
(**Minimierungsphase**)

Phase 2 transformiert das mehrstufige UND/ODER-Schaltnetz in Abhängigkeit der Gatterbiliothek in ein optimales CMOS-Schaltnetz. (**Optimierungsphase**)

Das Ergebnis der Synthese wird durch Vergleich mit der ursprünglichen Funktionsbeschreibung automatisch überprüft.
Das resultierende CMOS-Schaltnetz besteht aus NAND-Gattern, NOR-Gattern, Invertern und zweistufigen Komplexgattern mit bis zu 4 Eingängen (s. Bild 4.3 in Abschnitt 4.2).

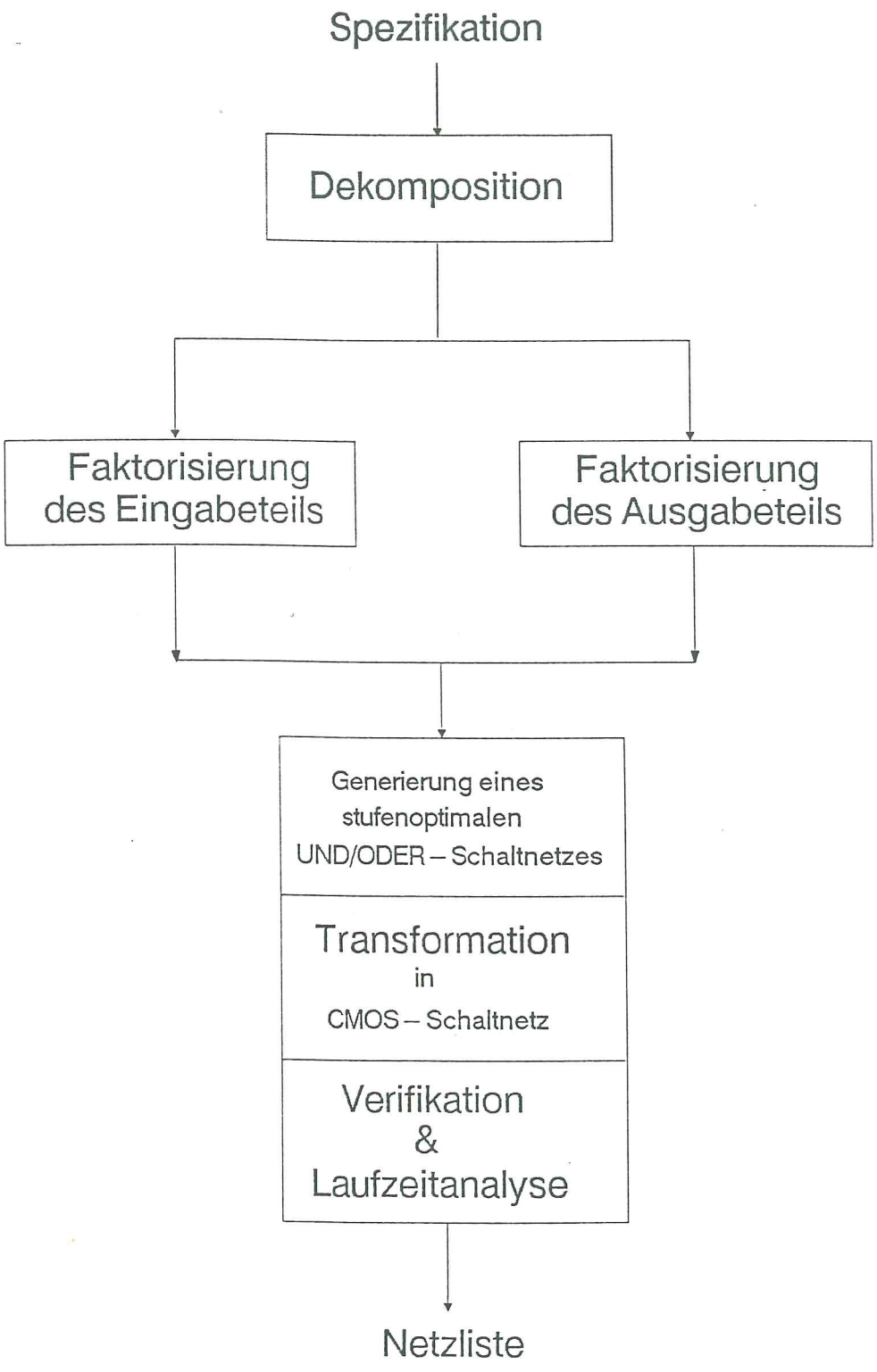


Bild 4.1: Mehrstufiger Schaltnetzentwurf - Entwurfsablauf CAR_MS

CAR_MS untersützt die Synthese von zwei verschiedenen Schaltnetzstrukturen (Bild 4.2):

Dekomposition: Durch Ausklammern von, möglicherweise mehrfach ausnutzbaren Teilfunktionen in disjunktiver Form entsteht eine baumförmige alternierende UND/ODER-Schaltnetzstruktur.

Faktorisierung: Durch Substitution von mehrfach vorkommenden konjunktiven bzw. disjunktiven Teiltermen entstehen vermaschte homogene UND- bzw. ODER-Strukturen.

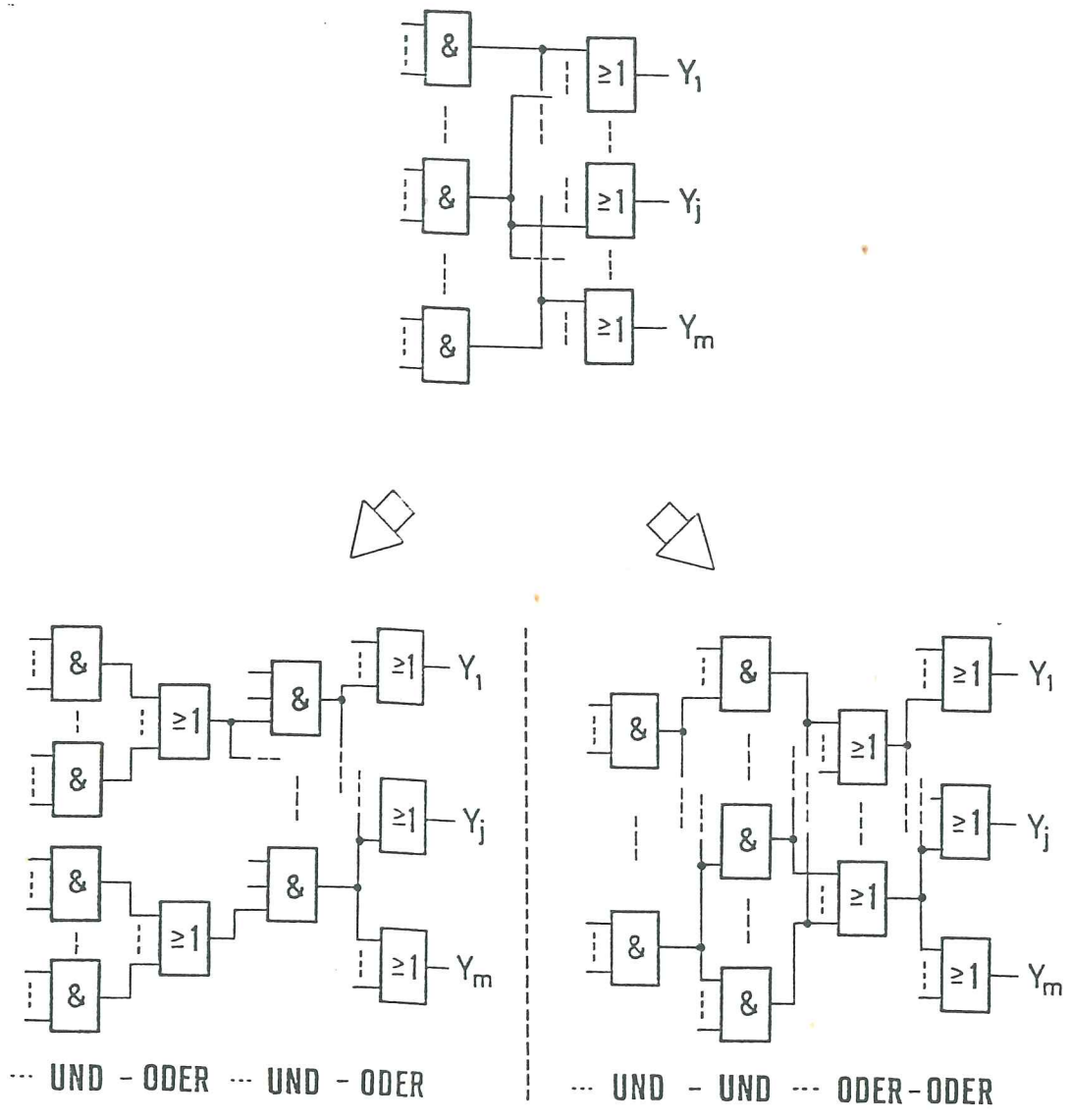


Bild 4.2: Mehrstufige Bündelschaltnetze

Beide Minimierungsmethoden können einzeln oder in der Reihenfolge "Dekomposition - Faktorisierung" miteinander kombiniert iterativ angewendet werden. Welche Vorgehensweise die bessere ist hängt letztlich von der ursprünglichen (zweistufigen) Darstellung der Schaltfunktion ab (Mehr dazu in Abschnitt 4.3).

4.2 Gatterbibliothek und Technologieparameter

Die CMOS-Gatterbibliothek enthält Inverter, NAND-Gatter, NOR-Gater und zweistufige Komplexgatter mit bis zu 4 Eingängen (siehe Bild 4.3).
Optional stehen Gatter mit zusätzlich invertierenden Ausgängen zur Verfügung (NAND/AND- und NOR/OR-Gatter, siehe Bild 4.4).
Die maximale Fan-Out-Belastung der Gatter beträgt 5. Die Inverter I1, I2, I3 und I4 haben ein maximales Fan-Out von 5, 10, 15 und 20.

Die charakteristischen Daten der Gatter sind in der ASCII-Datei GATLIB.DAT abgespeichert und können darin vom Entwerfer verändert werden. GATLIB.DAT ist in zwei Abschnitte gegliedert:

Im ersten Abschnitt stehen die Namen der Gatter und die Namen der Gatterein- und ausgänge. In der linken Spalte sind die programminternen symbolischen Namen aufgeführt. In die restlichen Spalten kann der Entwerfer die entsprechenden Namen der Gatter seiner Bibliothek eintragen. Zu beachten ist: 1) ein Name darf aus max. 7 Zeichen bestehen. 2) es werden genau 7 Zeichen als String eingelesen.

Im zweiten Abschnitt der Datei GATLIB.DAT sind die Gatterlaufzeiten tabellarisch in Abhängigkeit der Fan-Out-Belastung angegeben. In der letzten Spalte dieser Tabelle wird der Flächenbedarf des entsprechenden Gatters eingetragen.
Die Gatter, die optional zur Verfügung stehen (NAAN2 ... NOOR4), werden an dieser Stelle gekennzeichnet: ein negativer Flächenbedarf bedeutet, daß das entsprechende Gatter für den Entwurf nicht verwendet werden soll.

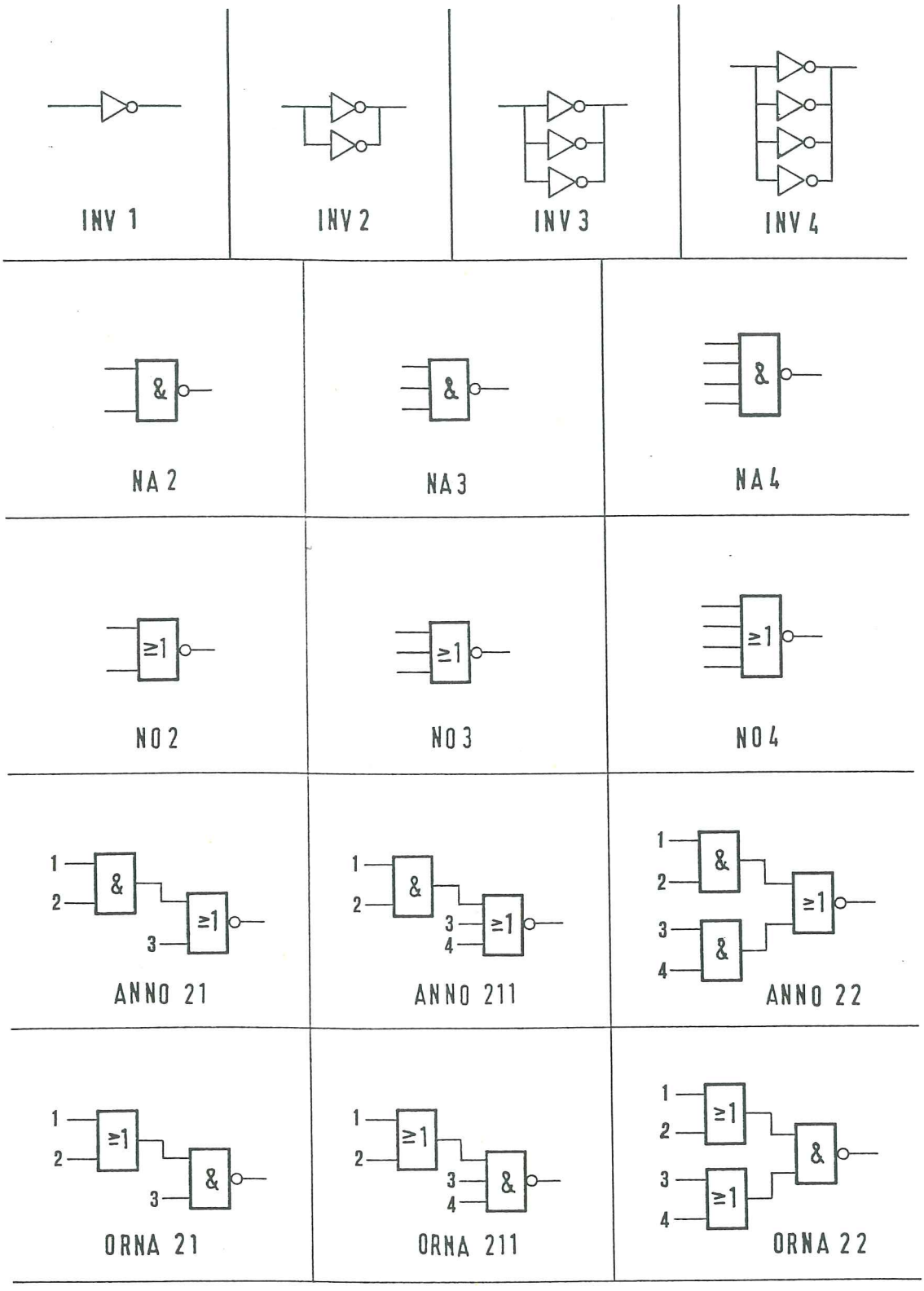


Bild 4.3: CARLOS Gatterbibliothek

Für die Synthese von sequentiellen Schaltungen stehen zwei Flipflop-Typen zur Verfügung: **D-Flipflops** mit CLEAR (DFFC) bzw. mit PRESET (DFFP) und **JK-Flipflops** mit CLEAR (JKFFC) bzw. mit PRESET (JKFFP). Die Namen der Flipflops und deren Anschlüsse sind programmintern festgelegt und nicht veränderbar (s. Tabelle 4.1).

C	CK	D	Q	NQ
1	-	-	0	1
0	0	-	Q^-	NQ^-
0	0-1	0	0	1
0	0-1	1	1	0

DFFC

P	CK	D	Q	NQ
1	-	-	1	0
0	0	-	Q^-	NQ^-
0	0-1	0	0	1
0	0-1	1	1	0

DFFP

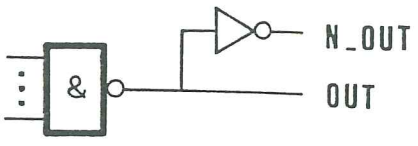
C	CK	J	K	Q	NQ
1	-	-	-	0	1
0	0	-	-	Q^-	NQ^-
0	0-1	0	0	Q^-	NQ^-
0	0-1	0	1	1	0
0	0-1	1	0	0	1
0	0-1	1	1	NQ^-	Q^-

JKFFC

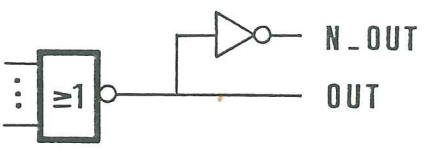
P	CK	J	K	Q	NQ
1	-	-	-	1	0
0	0	-	-	Q^-	NQ^-
0	0-1	0	0	Q^-	NQ^-
0	0-1	0	1	1	0
0	0-1	1	0	0	1
0	0-1	1	1	NQ^-	Q^-

JKFFP

Tabelle 4.1: Flipflop-Funktionstabellen



NAAN 2...4



NOOR 2...4

Bild 4.4: Optional zur Verfügung stehende Gatter

Inhalt der Datei **GATLIB.DAT**

```
* Elements of Gate-Library
*
*                    |          Name of Input          | Output
* Gate     | Name     |   1   |   2   |   3   |   4   |  Name
=================================================================
> I1       | INV1     | IN    |       |       |       | OUT    |
> I2       | INV2     | IN    |       |       |       | OUT    |
> I3       | INV3     | IN    |       |       |       | OUT    |
> I4       | INV4     | IN    |       |       |       | OUT    |
> NA2      | NAND2    | I1    | I2    |       |       | OUT    |
> NA3      | NAND3    | I1    | I2    | I3    |       | OUT    |
> NA4      | NAND4    | I1    | I2    | I3    | I4    | OUT    |
> NO2      | NOR2     | I1    | I2    |       |       | OUT    |
> NO3      | NOR3     | I1    | I2    | I3    |       | OUT    |
> NO4      | NOR4     | I1    | I2    | I3    | I4    | OUT    |
> AN2      | AND2     | I1    | I2    |       |       | OUT    |
> OR2      | OR2      | I1    | I2    | I3    |       | OUT    |
> ANNO21   | ANNO21   | IA1   | IA2   | IB    |       | OUT    |
> ANNO22   | ANNO22   | IA1   | IA2   | IB1   | IB2   | OUT    |
> ANNO211  | ANNO211  | IA1   | IA2   | IB    | IC    | OUT    |
> ORNA21   | ORNA21   | IA1   | IA2   | IB    |       | OUT    |
> ORNA22   | ORNA22   | IA1   | IA2   | IB1   | IB2   | OUT    |
> ORNA211  | ORNA211  | IA1   | IA2   | IB    | IC    | OUT    |
> NAAN2    | NAAN2    | I1    | I2    |       |       | OUT    | N_OUT
> NAAN3    | NAAN3    | I1    | I2    | I3    |       | OUT    | N_OUT
> NAAN4    | NAAN4    | I1    | I2    | I3    | I4    | OUT    | N_OUT
> NOOR2    | NOOR2    | I1    | I2    |       |       | OUT    | N_OUT
> NOOR3    | NAAN3    | I1    | I2    | I3    |       | OUT    | N_OUT
> NOOR4    | NOOR4    | I1    | I2    | I3    | I4    | OUT    | N_OUT
*
*
```

(Fortsetzung nächste Seite)

*
* **Worst Case Propagation Delay Times**
*
* | Propagation Delay Time in [ns]
* | Max [tpHL(Fan-Out), tpLH(Fan-Out)]

* Gate	Fan-Out					*
*	1	2	3	4	5	Area
> I1	1.556	2.012	2.468	2.924	3.380	0.5
> I2	1.316	1.532	1.748	1.964	2.130	1.0
> I3	1.244	1.388	1.532	1.676	1.820	1.5
> I4	1.220	1.340	1.460	1.580	1.700	2.0
> NA2	1.808	2.216	2.624	3.032	3.560	1.0
> NA3	2.100	2.700	3.300	3.900	4.500	1.5
> NA4	2.772	3.444	4.116	4.788	5.460	2.0
> NO2	2.676	3.252	3.828	4.404	4.980	1.0
> NO3	3.544	4.288	5.032	5.776	6.520	1.5
> NO4	4.460	5.420	6.380	7.340	8.300	2.0
> AN2	2.012	2.324	2.636	2.948	3.260	1.5
> OR2	2.788	3.076	3.364	3.652	3.940	1.5
> ANNO21	2.728	3.256	3.784	4.317	4.840	1.5
> ANNO22	2.656	3.112	3.568	4.024	4.480	2.0
> ANNO211	3.144	3.888	4.632	5.376	6.120	2.0
> ORNA21	2.500	3.100	3.700	4.300	4.900	1.5
> ORNA22	2.600	3.200	3.800	4.400	5.000	2.0
> ORNA211	3.352	3.904	4.456	5.008	5.560	2.0
> NAAN2	2.040	2.380	2.720	3.140	3.600	1.5
> (N_OUT)	1.960	2.220	2.480	2.740	3.000	
> NAAN3	2.500	3.000	3.500	4.000	4.500	2.0
> (N_OUT)	2.600	2.900	3.200	3.500	3.800	
> NAAN4	3.260	3.820	4.380	4.940	5.500	2.5
> (N_OUT)	3.400	3.700	4.000	4.300	4.600	
> NOOR2	3.080	3.560	4.040	4.520	5.000	1.5
> (N_OUT)	2.800	3.100	3.400	3.700	4.000	
> NOOR3	4.020	4.640	5.260	5.880	6.500	2.0
> (N_OUT)	3.500	3.800	4.100	4.400	4.700	
> NOOR4	5.100	5.900	6.700	7.500	8.300	2.5
> (N_OUT)	4.600	4.940	5.260	5.580	5.900	

4.3 Steuerung der Synthese

Durch Eingabe von Optionen hat der Entwerfer eine Vielzahl von Möglichkeiten den Entwurfsprozeß zu steuern.

Für die Dekomposition und die Faktorisierung stehen dem Entwerfer insgesamt fünf verschiedene Algorithmen zur Verfügung, die in Abhängigkeit ihrer Strategie mehr oder weniger Rechenzeit benötigen und damit mehr oder weniger optimale Ergebnisse liefern.
Die Auswahl reicht vom sehr schnellen Greedy-Algorithmus bis zum global, unter Zeitvorgabe optimierendem Algorithmus.

Durch Eingabe der folgenden Optionen ist die Güte des Schaltnetzes (Tradeoff zwischen Laufzeit und Flächenbedarf), die Struktur des Schaltnetzes und die Entwurfzeit beeinflußbar.

***OPTIONS**

Optionen zur Beeinflussung der Synthese-Strategie

NO_DEC - es erfolgt keine Dekomposition, u. U. günstig bei Bündeltabellen mit hoher Ausnutzung der Terme. Kann sich aber ungünstig auf die Verzögerungszeit des Schaltnetzes auswirken.

NO_FAC - es erfolgt keine Faktorisierung. Das resultierende Schaltnetz ist sehr schnell, benötigt aber viel Chipfläche.

NO_MAP - das UND/ODER-Schaltnetz wird nicht transformiert. Das Ergebnis kann auf eine andere Weise weiterverarbeitet werden.

Optionen zur Wahl des Optimierungskriteriums

MIN_NGA - Minimierung der Anzahl der Gatteräquivalente (= Gatter mit max. 2 Eingängen)

MIN_NCO - Minimierung der Anzahl der Verbindungen im Schaltnetz

Optionen zur Auswahl von Algorithmen

FAST_DEC - Dekomposition mit schneller heuristischer Auswahl.

GLOB_DEC - Globale Dekomposition; benötigt mehr Zeit und führt zu besseren Ergebnisse. Für das Auswahlproblem kann die Zeit (in Sek.) vorgegeben werden (Default: 5 Min/Iteration). Format: GLOB_DEC/t=100.

FAST_FAC - superschnelle Faktorisierung mit guten Ergebnissen

BEST_FAC - weniger heuristische Faktorisierungs-Strategie; Schaltnetze benötigen in der Regel weniger Fläche und weniger Laufzeit (im Vergleich zum FAST_FAC-Modus).

GLOB_FAC - Globale Faktorisierung; erzielt die besten Ergebnisse unter Zeitvorgabe (in Sek.) für das Auswahlproblem (Default: 5 Min/Iteration). Format: GLOB_FAC/t=100).

Optionen zur Beeinflussung der Stufenzahl

NIT0 = X - Dekompositionsphase: Es werden **X** Iterationen durchgeführt: Es entstehen maximal 2X zusätzliche Schaltnetzstufen (Default: keine Beschränkung).

NIT1 = Y - Faktorisierungsphase: Zur Optimierung des Eingabeteils werden **Y** Iterationen durchgeführt: Es entstehen dabei maximal Y UND-Stufen (Default: keine Beschränkung).

NIT1 = Z - Faktorisierungsphase: Zur Optimierung des Ausgabeteils werden **Z** Iterationen durchgeführt: Es entstehen dabei maximal Z ODER-Stufen (Default: keine Beschränkung).

Wenn der Entwerfer keine Optionen angibt, gelten die folgenden Voreinstellungen:

MIN_NGA, FAST_DEC, BEST_FAC, Anz. der Iterationen beliebig

Tips für den Schaltnetzentwurf:

Die Wahl der Optionen hängt im wesentlichen vom Entwurfsziel (Schaltnetz mit geringem Flächenbedarf, mit geringer Laufzeit, oder Kompromiß) und von der zweistufigen Darstellung der zu realisierenden Schaltfunktion ab.

Die **Laufzeit** nimmt erfahrungsgemäß unabhängig von der Entwurfsstrategie (Dekomposition und/oder Faktorisierung) mit der Stufenzahl des Schaltnetzes proportional zu. Sie kann durch Beschränkung der Anzahl (nit0,nit1 und nit2) der Iterationen klein gehalten werden (Eine untere Grenze für die Laufzeit erhält man durch eine 1:1-Transformation der Strukturtabelle (Optionen: NO_DEC und NO_FAC). Geringfügig beeinflussen läßt sich die Laufzeit auch durch Vorgabe des Optimierungskriteriums: Option MIN_NCO liefert Schaltnetze, deren Laufzeit oft geringer (aber nie höher) ist als mit MIN_NGA synthetisierte Schaltnetze.
Geringe Laufzeiten (bei geringem Flächenbedarf) erzeugt die Option GLOB_FAC gegenüber den Optionen FAST_FAC und BEST_FAC.

Durch geeignete Wahl der zweistufigen Form läßt sich die Laufzeit ebenfalls günstig beeinflussen: Die Erfahrung zeigt, daß einzeln minimierte Funktionen eines Bündels (Option SINGLE für CAR_TM) hinsichtlich der Laufzeiten einen günstigeren Ausgangspunkt darstellen als im Bündel minimierte Funktionen.

Beispiel: Schaltfunktion f, 14 Eingänge, 5 Ausgänge

a) Bündelminimierung: 60 Produktterme, 360 Gatteräquivalente
   Mehrstufige Synthese (GLOB_DEC, GLOB_FAC):
   200 Grundzellen, 35 ns
b) Einzelminimierung: 86 Produktterme, 500 Gatteräquivalente
   Mehrstufige Synthese (GLOB_DEC,GLOB_FAC):
   240 Grundzellen, 25 ns

Wichtig: Einzeln minimierte Funktionen sollten grundsätzlich dekomponiert werden (also: keine Option NO_DEC).

Wenn der minimale **Flächenbedarf** das Hauptziel des Entwurfs ist, sollte

1) die Anzahl der Iterationen nicht beschränkt werden (Voreinstellung) und
2) als Optimierungskriterium die Anzahl der Gatteräquivalente gewählt werden (ebenfalls Voreinstellung).

Die Dekomposition sollte bei einzeln minimierten Funktionen auf jeden Fall angewendet werden. Bei Bündelfunktionen mit starker Mehrfachausnutzung der Produktterme kann die Dekomposition zu einem hinsichtlich des Flächenbedarfs schlechteren Ergebnis führen (Bitte ausprobieren, die Programme sind schnell genug!).

Bei der Faktorisierung nimmt die Güte der Ergebnisse in der Reihenfolge der Optionen FAST_FAC, BEST_FAC und GLOB_FAC zu (und ebenso die Rechenzeit).

4.4 Ausgabe von CAR_MS

4.4.1 Netzliste

Die von CAR_MS erzeugte Netzliste wird in einer ASCII-Datei abgespeichert.
Zu Beginn der Netzliste steht die Identifikation der Schaltung. Alle weiteren Zeilen beginnen jeweils mit einer Schlüsselnummer in der ersten Spalte. Die Schlüsselnummern und deren Bedeutungen sind in Tabelle 4.2 aufgeführt.

Schlüsselnummer	Bedeutung
2	Zweipunktverbindung (intern)
3	Zweipunktverbindung mit externem Anschluß
5	Gatterstatistik
6	Beginn der Beschreibung eines Netzes
9	Netzname (bei Netzen mit externem Anschluß)

Tabelle 4.2: Bedeutung der Schlüsselnummern in der Netzliste

Die Beschreibung der Schaltungsstruktur erfolgt durch Angabe von Netzen, wobei ein Netz mit n Zweigen durch n Zweipunktverbindungen dargestellt wird (entsprechend n Zeilen in der Netzliste). Vor der Auflistung der Netze, werden die verwendeten Gatter- und Flipfloptypen und deren Häufigkeit aufgeführt (Zeilen mit Schlüsselnumer 5).

Zeilen mit der Schlüsselnummer 5 haben das folgende Format:

5 0 x <G>

x : Anzahl von Bausteinen vom Typ <G>
<G> : Name des Gatter- bzw. Flipfloptyps

Die Zeilen mit der Schlüsselnummer 5 folgen unmittelbar der Kopfzeile.

Das Format für ein Netz mit n Zweigen ist wie folgt:

Zeile 1: **6 0 y .** (definiert den Anfang der Netzbeschreibung)

y = 0: internes Netz
1: Netz mit externem Anschluß

Zeile 2: **9 -1 z<S>** (nur bei Netzen mit externem Anschluß, y=1)

z = I: Eingangssignal
= O: Ausgangssignal
<S> : Name des Signals

Für jeden der n Zweige folgt eine Zeile. Es wird unterschieden zwischen internen Verbindungen und Verbindungen mit externem Anschluß.

Zeile i: **2 -1 k z<A><G>** (internes Netz)

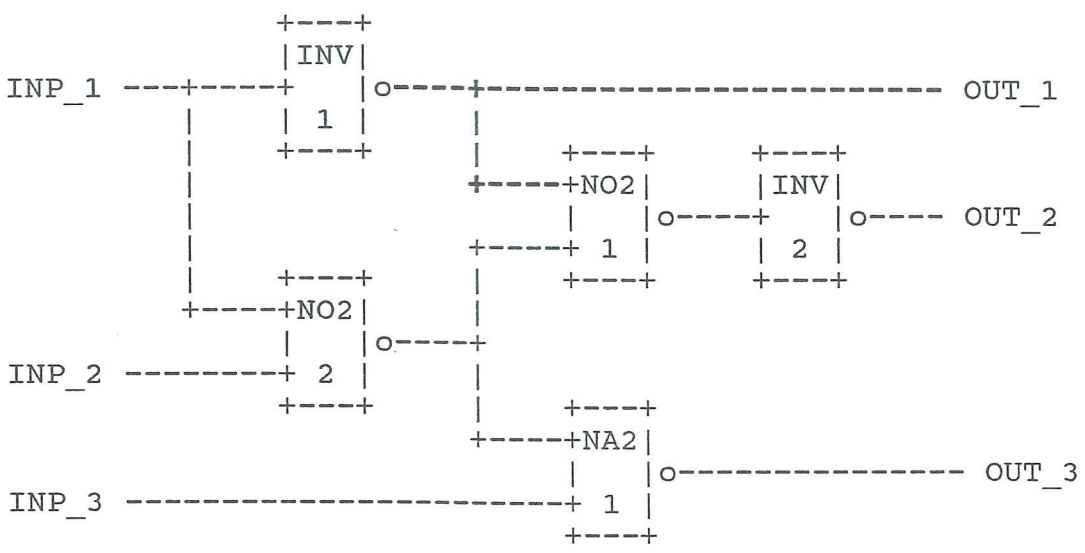
k : Gatternummer
z = I: Eingangssignal
= O: Ausgangssignal
<A> : Name des Anschlusses
<G> : Name des Gatter- bzw. Flipfloptyps

Zeile j: **3 -1 0 z<S>** (externe Verbindung)

z = I: Eingangssignal
= O: Ausgangssignal
<S> : Name des Signals

In der Netzliste erscheinen die Namen der Eingangs- und Ausgangssignale, die unter den Schlüsselwörtern *X-NAMES und *Y-NAMES in der Eingabedatei vereinbart wurden. (Zustands- und Folgezustandssignale sind interne Signale!)
Die Nummerierung der Gatter erfolgt nach Gattertypen getrennt, d. h. ein Gatter in der Netzliste wird erst durch Angabe von Gatternummer und Gattertyp eindeutig bestimmmt.

Abschließend ein Beispiel:



Netzliste:

```
DEMO 1
5 0 2        INV
5 0 2        NO2
5 0 1        NA2
6 0 1 .
9 -1 OOUT_1
3 -1 0 OOUT_1
2 -1 1 OOUT      INV
2 -1 1 II1       NO2
6 0 1 .
9 -1 OOUT_2
3 -1 0 OOUT_2
2 -1 2 OOUT      INV
6 0 0 .
2 -1 1 OOUT      NO2
2 -1 2 IIN       INV
6 0 0 .
2 -1 2 OOUT      NO2
2 -1 1 II2       NO2
2 -1 1 II1       NA2
6 0 1 .
9 -1 OOUT_3
3 -1 0 OOUT_3
2 -1 1 OOUT      NA2
6 0 1 .
9 -1 IINP_1
3 -1 0 IINP_1
2 -1 1 IIN       INV
2 -1 2 II1       NO2
6 0 1 .
9 -1 IINP_2
3 -1 0 IINP_2
2 -1 2 II2       NO2
6 0 1 .
9 -1 IINP_3
3 -1 0 IINP_3
2 -1 1 II2       NA2
```

4.4.2 Statistische Daten

Das Programm CAR_MS legt eine ASCII-Datei an, in der statistische Daten über den Schaltungsentwurf abgespeichert werden. Diese Daten sind in drei Abschnitte unterteilt:

I. Circuit Identification and Parameters

Der erste Abschnitt enthält Informationen über die zu realisierende Schaltung: Namen, Anzahl von Eingangs- und Ausgangs- und Zustandsvariablen und Angaben über die Größe des zweistufigen Bündelschaltnetzes.

II. Multilevel AND/OR-Circuit Characteristics

Hier findet man in tabellarischer Form charakteristische Daten über das mehrstufige UND/ODER-Schaltnetz. Für jedes interne Zwischensignal und Ausgangsignal ist die Stufenzahl (level) und die Fan-Out-Belastung angegeben. Bei der Interpretation der Stufenzahl ist zu beachten, daß das UND/ODER-Schaltnetz ausschließlich aus Gattern mit zwei Eingängen besteht.

III. Multilevel CMOS-Circuit Characteristics

Dieser Bereich beschreibt Charakteristika des synthetisierten CMOS-Schaltnetzes: Gatterstatistik, Flächenbedarf der Schaltung, Verzögerungszeiten (Worst-Case) für alle Schaltnetze und die Fan-Out-Belastung der Ein- und Ausgangssignale der Schaltung.

4.5 Fehlermeldungen

CAR_MS extrahiert aus dem programminternen mehrstufigen Schaltnetzgraph die korrespondierende Schaltfunktion in disjunktiver Form und vergleicht diese mit der Spezifikation des zweistufigen Schaltnetzes, das durch die Strukturtabelle in der Eingabedatei <NAME>.STR beschrieben wird. Treten Widersprüche auf, erscheint folgende Fehlermeldung auf dem Bildschirm:

"*** ERROR ****** cube ... is not covered".**

Der Syntheseablauf wird abgebrochen. Diese Fehlermeldung deutet auf einen Programmfehler hin. Bitte informieren Sie umgehend den Wartungsdienst.

5. Literatur zu CARLOS

Eine Übersicht (Stand: März 86) findet man in

[1] H.-J. Mathony; U. G. Baitinger;
"CARLOS: An Automated Multilevel Logic Design System for CMOS Semicustom Integrated Circuits"
IEEE Trans. on Computer-Aided-Design, Vol. 7, No. 3, 1988

Die neueren Algorithmen für den mehrstufigen Entwurf sind beschrieben in

[2] H.-J. Mathony; U. G. Baitinger;
"Fast Efficient Algorithms for the Factoring of Multiple Output Logic Functions",
IEEE International Symposium on Circuits and Systems Proceedings, 1988, p. 1851-1854

und in

[3] H.-J. Mathony;
"Algorithmische Entwurfsverfahren für zwei- und mehrstufige Schaltnetze", Dissertation, Uni Karlsruhe 1987.
VDI-Verlag, Reihe 9

Die Algorithmen für die zweistufige Minimierung sind in [3], [4] und [5] (neuester Stand) beschrieben.

[4] H. Dücker;
"Implementierung und Untersuchung von Algorithmen für die Minimierung von zweistufigen Bündelschaltnetzen",
Diplomarbeit ID-509, ITIV, Univ. Karlsruhe, 1987

[5] H.-J. Mathony;
"A Universal Logic Design Algorithm and its Application to the Synthesis of Two-Level Circuits" (Technical Report 1988)

Eingabe für Codierer

```
*IDENTIFICATION
 Trommelsteuerung
*DECLARATIONES
 INITIAL-STATES  = 1
 X-VARIABLES     = 9
 Y-VARIABLES     =12
*FLIPFLOPS
 D-FLIPFLOPS
*X-NAMES
 Reset = 1,  Start = 2,  RDWR = 3,  CNT0 = 4,  CNT1 = 5,  CNT23 = 6,
 ADMTCH = 7,  EOB = 8,  PARITY = 9
*Y-NAMES
 ADEN = 1,  BRA = 2,  BRB = 3,  SRA = 4,  SRB = 5,  PARA = 6,
 PARB = 7,  ENPAR = 8,  LS = 9,  BUSY = 10,  DRQ = 11,  PERROR = 12
*OPTIONS
 FIGARO, TERME
*FLOWTABLE

; kommentarzeile die 1.

;   kommentar der 2.
S 1, X1--------, F 1, Y--------0000;
S 1, X00-------, F 1, Y---00---0000;
S 1, X011------, F 2, Y1101000-0000;
S 1, X010------, F 8, Y110--00-0000;
S 2, X1--------, F 1, Y--------0000;
S 2, X0--------, F 3, Y0000010-0100;
S 3, X1--------, F 1, Y--------0000;
S 3, X0--1--1--, F 4, Y000010101100;
S 3, X0--0-----, F 3, Y0000010-0100;
S 3, X0-----0--, F 3, Y0000010-0100;
S 4, X1--------, F 1, Y--------0000;
S 4, X0----1-1-, F 6, Y000--0101100;
S 4, X0----1-0-, F 5, Y000--0101100;
S 4, X0----0---, F 4, Y000010101100;
S 5, X1--------, F 1, Y--------0000;
S 5, X0--------, F 4, Y001100011110;
S 6, X1--------, F 1, Y--------0000;
S 6, X0--------, F 7, Y-------11100;
S 7, X1--------, F 1, Y--------0000;
S 7, X01-------, F 7, Y---00---0000;
S 7, X00-------, F 1, Y---00---0000;
S 8, X1--------, F 1, Y--------0000;
S 8, X0--------, F 9, Y000--10-0100;
S 9, X1--------, F 1, Y--------0000;
S 9, X0--------, F10, Y000--10-0100;
S10, X1--------, F 1, Y--------0000;
S10, X0---0----, F10, Y000--10-0100;
S10, X0-----0--, F10, Y000--10-0100;
S10, X0---1-1--, F11, Y0000111-0100;
S11, X1--------, F 1, Y--------0000;
S11, X0--1---10, F 7, Y0010011-0100;
S11, X0--1----1, F13, Y0010011-0100;
S11, X0--1---00, F12, Y0010011-0100;
S11, X0--0-----, F11, Y0000111-0100;
S12, X1--------, F 1, Y--------0000;
S12, X0--------, F11, Y0000011-0110;
S13, X1--------, F 1, Y--------0000;
S13, X0--------, F13, Y00000---0001;
*END
```

```
*IDENTIFICATION
  Trommelsteuerung
  TERM
*DECLARATIONES
  X-VARIABLES      =  9
  Y-VARIABLES      =  10
  QX-VARIABLES     =  4
  QY-VARIABLES     =  4
*FLIPFLOPS
  D-FLIPFLOPS
*X-NAMES
Reset =5,  Start =6,  RDWR =7,  CNT0 =8,  CNT1 =9,  CNT23 =10,
ADMTCH =11,  EOB =12,  PARITY =13
*Y-NAMES
ADEN =5,  BRA =6,  BRB =7,  SRA =8,  SRB =9,  PARA =10,
PARB =11,  ENPAR =12,  LS =13,  BUSY =14,  DRQ =15,  PERROR =16
*Y-IDENT
  Y6: Y5,
  Y7: Y12,
*PRESET/RESET
  0000
*FUNCTIONTABLE
   0001--------, 0000------0000
  000000--------, 0000--00--0000
  0000011------, 011010100000000
  0000010------, 011110--000000
  01101--------, 0000------0000
  01100--------, 101000001001000
  10101--------, 0000------0000
  10100--1--1--, 101100010111000
  10100--0-----, 101000001001000
  10100-----0--, 101000001001000
  10111--------, 0000------0000
  10110----1-1-, 100000--011100
  10110----1-0-, 100100--011100
  10110----0---, 101100010111000
  10011--------, 0000------0000
  10010--------, 101101100011100
  10001--------, 0000------0000
  10000--------, 0100-1----1100
  01001--------, 0000------0000
   10001-------, 0100--00--0000
  010000-------, 0000--00--0000
  01111--------, 0000------0000
  01110--------, 000100--100100
  00011--------, 0000------0000
  00010--------, 111000--100100
  11101--------, 0000------0000
  11100---0----, 111000--100100
  11100-----0--, 111000--100100
  11100---1-1--, 111100011101000
  11111--------, 0000------0000
  11110--1---10, 010001001101000
  11110--1----1, 110101001101000
  11110--1---00, 110001001101000
  11110--0-----, 111100011101000
  11001--------, 0000------0000
  11000--------, 111100001101100
  11011--------, 0000------0000
  11010--------, 1101Ï0000--0001

*END
```

Ausgabe des Codierers für CARLOS

CPU-TIME USED 4.800

Zwischenformat für CARLOS

```
*IDENTIFICATION
  Trommelsteuerung
 TERM
*DECLARATIONES
 X-VARIABLES      =  9
 Y-VARIABLES      =  10
 QX-VARIABLES  =  4
 QY-VARIABLES  =  4
*FLLIPFLOPS
 D-FLIPFLOPS
*X-NAMES
Reset =5,  Start =6,  RDWR =7,  CNT0 =8,  CNT1 =9,  CNT23 =10,
ADMTCH =11,  EOB =12,  PARITY =13
*Y-NAMES
ADEN =5,  BRA =6,  BRB =7,  SRA =8,  SRB =9,  PARA =10,
PARB =11,  ENPAR =12,  LS =13,  BUSY =14,  DRQ =15,  PERROR =16
*Y-IDENT
 Y6: Y5,
 Y7: Y12,
*PRESET/RESET
 0000
*STRUCTURE-TABLE
 -110  0-- --- ---        1010 00 001 001 000 000
 -011  0-- --- ---        1000 00 010 111 000 000
 0-01  0-- --- ---        1110 00 001 001 000 000
 1111  0-- 0-- ---        1011 00 010 101 000 000
 1000  0-- --- ---        0100 01 000 011 000 000
 0111  0-- --- ---        0001 00 001 001 000 000
 1111  0-- 1-- ---        0000 01 000 101 000 000
 --10  0-- 0-- ---        1010 00 001 001 000 000
 00-0  01- --- ---        0010 10 100 000 000 000
 10-1  0-- --0 ---        0011 00 000 000 000 000
 11--  0-- --- ---        0100 00 001 000 000 000
 --10  0-- --- 0--        1010 00 001 001 000 000
 1110  0-- -1- 1--        0001 00 010 100 000 000
 1111  0-- 1-- --1        1001 00 000 000 000 000
 --00  01- --- ---        0100 00 000 000 000 000
 1011  0-- --1 -0-        0001 00 000 000 000 000
 1111  0-- 1-- -00        1000 00 000 000 000 000
 1001  0-- --- ---        1011 01 100 011 100 000
 1100  0-- --- ---        1011 00 000 101 100 000
  101  0-- --- ---        1001 00 000 000 010 000
 -010  0-- 1-- 1--        1011 00 010 111 000 000
 00-0  010 --- ---        0001 00 000 000 000 000
*END
   CPU-TIME USED  4.800
```

Eingabe für Codierer

```
*IDENTIFIKATION
 TAST
*DECLARATIONS
 INITIAL-STATE = 1
 X-VARIABLE   = 8
 Y-VARIABLE   = 10
*FLIPFLOPS
 D-FLIPFLOPS
*X-NAMES
 X1 = 1,X2 = 2,X3 = 3,X4 = 4,X5 = 5,X6 = 6,X7 = 7,X8 = 8;
*Y-NAMES
 Y1 = 1,Y2 = 2,Y3 = 3,Y4 = 4,Y5 = 5,Y6 = 6,Y7 = 7,Y8 = 8,
 Y9 = 9,Y10 = 10;
*OPTION
 LOGIC, LITERALE
*FLOWTABLE

S 1, X--------, F 2, Y----------;
S 2, X11------, F 3, Y----------;
S 2, X0-------, F 2, Y----------;
S 2, X-0------, F 2, Y----------;
S 3, X--10----, F 2, Y----------;
S 3, X--11----, F 4, Y1000000000;
  3, X--0-----, F 2, Y----------;
S 4, X--------, F 5, Y----------;
S 5, X----1---, F 4, Y1---------;
S 5, X----0---, F 6, Y0100000000;
S 6, X-----0--, F 6, Y0100000000;
S 6, X-----1--, F 7, Y0111110000;
S 7, X--------, F 8, Y0111111100;
S 8, X------00, F 8, Y0111111100;
S 8, X-------1, F 9, Y0010000010;
S 8, X------10, F16, Y0010000001;
S 9, X11------, F10, Y----------;
S 9, X0-------, F 9, Y----------;
S 9, X-0------, F 9, Y----------;
S10, X--0-----, F 2, Y----------;
S10, X--10----, F 9, Y----------;
S10, X--11----, F11, Y1000000000;
S11, X--------, F12, Y----------;
S12, X----0---, F13, Y0100000000;
S12, X----1---, F11, Y1000000000;
  3, X-----0--, F13, Y0100000000;
S13, X-----1--, F14, Y0111100000;
S14, X--------, F15, Y0111101100;
S15, X------00, F15, Y0111101100;
S15, X-------1, F 9, Y0001000010;
S15, X------10, F16, Y0010000001;
S16, X-------0, F16, Y----------;
S16, X-------1, F 2, Y----------;
*END
```

Ausgabe des Codierers für Logic

```
*IDENTIFIKATION
  TAST
 LITERALE
*DECLARATIONS
 X-VARIABLE  =  12
 Y-VARIABLE  =  13
*X-NAMES
Q1 =  1, Q2 =  2, Q3 =  3, Q4 = 4,
X1 =5,X2 =6,X3 =7,X4 =8,X5 =9,X6 =10,X7 =11,X8 =12;
*Y-NAMES
Q1 =  1, Q2 =  2, Q3 =  3, Q4 = 4,
Y1 =5,Y2 =6,Y3 =7,Y4 =8,Y5 =9,Y6 =10,Y7 =11,Y8 =12,
 Y9 =13;
*RUN-CONTROL
 CPU-TIME = 600
 LISTING = LONG
*FUNCTIONTABLE
 1111--------: 0000---------;
 000011------: 0100---------;
 00000-------: 0000---------;
 0000-0------: 0000---------;
 0100--10----: 0000---------;
  100--11----: 0010100000000;
 0100--0-----: 0000---------;
 0010--------: 1100---------;
 1100----1---: 00101--------;
 1100----0---: 0110010000000;
 0110-----0--: 0110010000000;
 0110-----1--: 0111011111000;
 0111--------: 1001011111100;
 1001------00: 1001011111100;
 1001-------1: 0001001000010;
 1001------10: 0011001000001;
 000111------: 0101---------;
 00010-------: 0001---------;
 0001-0------: 0001---------;
 0101--0-----: 0000---------;
 0101--10----: 0001---------;
 0101--11----: 1000100000000;
 1000--------: 1101---------;
 1101----0---: 1010010000000;
  101----1---: 1000100000000;
 1010-----0--: 1010010000000;
 1010-----1--: 1110011110000;
 1110--------: 1011011110100;
 1011------00: 1011011110100;
 1011-------1: 0001000100010;
 1011------10: 0011001000001;
 0011-------0: 0011---------;
 0011-------1: 0000---------;
*PAL
 TYPE = HYPERPAL
*END

   CPU-TIME USED  4.350
```